NAS Technical Report
NAS-04-015
November, 2004

Automated Parameter Studies Using a Cartesian Method

Scott M. Murman[*]
ELORET
MS T27B
Moffett Field, CA 94035

Michael J. Aftosmis[†]
NASA Ames Research Center
MS T27B
Moffett Field, CA 94035

Marian Nemec[‡]
NASA Ames Research Center
MS T27B
Moffett Field, CA 94035

Abstract

A modular process for performing general parametric studies about an aerodynamic configuration using a Cartesian method is described. A novel part of this process is the automatic handling of general control surfaces deflections based upon simple, user-specified inputs. The article focuses on the use of aerodynamic databases in the design process. Database fly-through is used to develop and analyze guidance and control systems, and to evaluate performance data. Validation comparisons with experimental data and Navier-Stokes simulations are presented for the Langley Glide-Back Booster vehicle. Two example parametric databases with control surfaces deflections are presented: an autonomous Mars explorer aircraft which contains 4700 datapoints and two movable elevons, and the Space Shuttle launch vehicle in ascent configuration with gimbaling engine nozzles. The database for the Mars aircraft has been used to validate a generic neural-network control system, and trajectory simulations using the shuttle aerodynamic data are coupled with an optimization algorithm to develop a closed-loop feedback pitch controller.

[*]Senior Research Scientist, smurman@mail.arc.nasa.gov
[†]Research Scientist, aftosmis@nas.nasa.gov
[‡]NRC Research Associate, nemec@nas.nasa.gov

1 Introduction

Computational Fluid Dynamics (CFD) is now routinely used to analyze isolated points in a design space by performing steady-state computations at fixed flight conditions (Mach number, angle of attack, sideslip), for a fixed geometric configuration of interest. This isolated point analysis is typically performed using high fidelity methods at a handful of critical design points, e.g. a cruise or landing configuration, or a sample of points along a flight trajectory. Current research in CFD is aimed at extending the point analysis to rapidly analyze the entire design space with high-fidelity tools - varying both flight conditions and control surface deflections - in order to provide a broader picture of performance[1–4]. Such a database is used both in initial design to quickly estimate performance, and in final design to augment traditional methods such as wind tunnel tests and aerodynamic modeling. The ability to rapidly assimilate an accurate database of aerodynamic performance using CFD methods opens up new design possibilities for the engineer. The aerodynamic database can be used with six-degree-of-freedom (6-DOF) trajectory simulations coupled to guidance and control (G&C) systems. This ability to "fly" a design through an aerodynamic database in faster-than-real-time enables performance estimates for prototypes to be rapidly evaluated, and novel guidance and control (G&C) systems developed. The focus of the current work is the development of automated and efficient tools for building aerodynamic databases, and the use of controlled 6-DOF trajectories for evaluating vehicle designs using these databases.

A typical CFD aerodynamic database currently contains on the order of $10^4 - 10^6$ datapoints, depending upon the problem requirements. In order to manage such a large number of computations, automated tools are a necessity, not only for generating the data, but also harvesting meaningful results in a post process. The current work uses a Cartesian, embedded-boundary method[5] to automate the generation of a vehicle aerodynamic parameter study. The Cartesian method provides an efficient and robust mesh generation capability which can handle an arbitrarily-complex geometry description. Recently, a method to generate the water-tight surface triangulation required for Cartesian mesh generation directly from a CAD representation of the geometry has been developed[6]. This, combined with the Cartesian embedded-boundary method provides a robust and automatic mesh generation infrastructure which can be utilized through the design process. This meshing scheme is combined with a parallel, multi-level scheme for solving the steady-state Euler equations, either on shared- or distributed-memory architectures[7, 8].

In order to analyze the entire design space it is necessary to automate the analysis of control surface deflections, as well as changes to the velocity vector. A novel part of the current work is a general system of specifying and manipulating control surfaces using the Geometry Manipulation Protocol (GMP)[9]. This system automatically re-generates the appropriate rigid-body configurations from a geometry and control surface description. The allowable rigid-body motions are extremely general, so that no new application code is required to perform the geometry manipulation.

This paper first describes the software design of the modular infrastructure built to automatically perform parameter studies. This modular system is built around pre-existing stand-alone applications (mesh generator, flow solver, force/moment calculator, etc.), using scripts to provide a flexible "glue" between components. An overview of the 6-DOF trajectory algorithms for database fly-through is also presented. The last three sections present example aerodynamic databases and applications. First, a validation comparison with both experimental data and Navier-Stokes simulations is presented for the Langley Glide-Back Booster vehicle. Next, a database including 4700 datapoints for a prototype autonomous biomorphic explorer for Mars[10] is included. The Mars flyer design has an elevon on each wing to provide control power. This CFD database has been used to validate a neural-network G&C system for future flight tests. Finally, a sample database for the Space Shuttle Launch Vehicle (SSLV) in ascent configuration with gimbaling engine nozzles is presented. This database is used to develop a feedback control system by coupling the trajectory simulations with an optimizer module to provide optimal feedback gains. This optimized controller is compared to data from the STS-107 flight of the SSLV.

2 Component Infrastructure

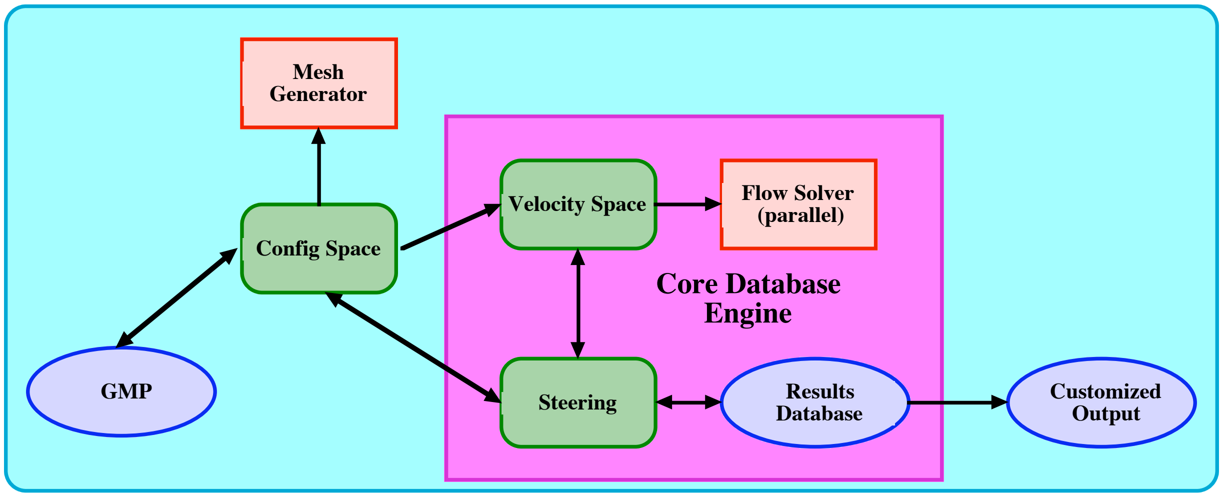


Figure 1: Schematic of the modular set of software components for performing an automated CFD parameter sweep. The stand-alone applications are shown in red, and the control scripts which glue the applications together in green. Third-party application data is in blue. The purple box contains the core components: added functionality simply wraps around this core.

Figure 1 shows a schematic of the modular set of software components used for building a CFD aerodynamic database. The stand-alone applications are shown in red, and the control scripts which glue the applications together in green. More details on the specifics of the infrastructure will be provided after a brief overview. Note that several low-level support scripts which are common to all of the tools are not described. The entire parameter space is decomposed into a two-level hierarchy. At the lowest level is a *velocity space* which contains the wind vector for each data point. This is characterized by the Mach number, angle of attack, and sideslip angle $(M_\infty, \alpha, \beta)$. The next higher level contains a *configuration space* which represents all of the possible geometric configurations (e.g. control surface deflections, δ_c) being tested. A single element of the configuration space contains all elements of the velocity space. The rationale for this decomposition is the modular re-use of components. Each element of the velocity space uses a fixed computational mesh, and is thus independent of the configuration space, which requires a modified geometry and mesh for each element. In other words, a parameter study can be computed using the velocity space in isolation, or as a lower-level to a broader configuration space, without requiring modification of the velocity space software components or architecture. The *steering software* links the configuration space, velocity space, and the results database into a coherent unit. The steering software monitors the results and determines which element of the parameter space to compute next, and this command is sent to the configuration space control script (or directly to the velocity space control script if only a single configuration is being examined). The configuration control then passes this to the velocity space. This hierarchy of command control continues through the process, with each control script being responsible for handling input and output from their isolated section of the process. This is similar to an object-oriented framework, though here a rigid object-oriented interface is not maintained.

Since the elements of the process infrastructure are independent, each can evolve in isolation. For example, the steering software can be as simplistic as iterating through a uniform parameter space following the matrix element ordering, or as complex as leveraging 3rd-party neural-network toolkits. This flexibility is key to providing a system which can grow as more experience is gained with building aerodynamic databases using CFD. Currently the steering software is intentionally straightforward: the parameter space is maintained as a regular Cartesian mesh and traversed sequentially. In order for more complex adaptive steering mechanisms to be utilized, a general unstructured data format and database interpolation is required (cf. Sec. 2.3.1). This generalization is left for future work after experience is gained using the current implementation.

The emphasis of the current work in on an automated process, hence the user interfaces are at the highest level of the component hierarchy. The analyst is charged with providing inputs to the system, and harvesting results, however the control of the process is the purview of the script system. This is necessary, as managing or steering tens of thousands of computations by hand is not practical or desirable. The understanding is that if more details are required about a certain isolated point, or set of points, the analyst will perform a separate point analysis at those critical points.

One implicit assumption with the implementation of the infrastructure outlined in Fig. 1 is that there are no conflicts for CPU resources between the isolated parts of the process. In other words, it is assumed that the lightweight processes of generating a mesh, or adding an entry into the results database, will not interfere with the more compute-intensive flow solver processes. In practice this is not a restrictive assumption. Most compute resources either provide a front-end machine, or isolate a CPU or set of CPUs, which are responsible for job scheduling, interactive terminal sessions, etc. Since the cost of mesh generation and post-processing is amortized over thousands of flow solver runs, and these tasks run concurrent with the flow solver, these lightweight processes do not adversely impact the overall parallel efficiency.

2.1 Velocity Space

The implementation of the velocity space control script is straightforward. The control script sets the velocity space parameters which are inputs for the flow solver, creates any required directory and file structures, and then executes the solver. After the solver finishes, the aerodynamic coefficients are returned as results of the control script to the process which executed the velocity space script. These coefficients are selected by the user, and can contain items such as hinge moments or bending moments, in addition to the 6 aerodynamic coefficients for the entire configuration $(C_A, C_Y, C_N, C_l, C_m, C_n)$. The flow solver can execute either on a single processor or in parallel. Filling a parameter space involves computing many essentially identical problems, hence the parallelism is usually exploited on this coarser-grained level, not at the finer level of the flow solver, however there is nothing which prohibits combining both fine- and coarse-grained parallel strategies. The Cartesian solver does not require case-specific inputs for each set of flow conditions, which greatly simplifies the velocity space sub-system. The solver contains a hierarchy of robustness levels, with each increasing level requiring more computing resources. The run strategy simply starts with the least robust scheme, and proceeds up the hierarchy if a computation is numerically unstable. At the highest level of robustness it is always possible to maintain numerical stability, though it is still possible to generate spurious results, for example when computing an inherently unsteady problem with a steady-state method. In practice these pathological cases are relatively easy to filter, for example by monitoring convergence in residuals and computed aerodynamic loads. The velocity space control script is responsible for creating the flow solver runtime environment. Several interfaces are provided, including a simple interactive runtime process, job management directly through the portable batch system (PBS)[11], and an interface to the web-based AeroDB job control software[3]. Again, since the velocity-space control scripts are self-contained they can evolve along with new runtime environments without effecting the remainder of the process.

2.2 Configuration Space

One powerful feature of the Cartesian method is the ability to automatically create a computational mesh about varying geometric configurations, as has been demonstrated for computing bodies in relative motion[12, 13] and aerodynamic shape optimization[14]. Moving control surfaces within a static configuration space also requires the ability to "re-mesh" changing geometric configurations. The complete set of control surfaces and their possible deflections, or orientations, makes up the configuration space. This configuration space is described using the ConfigSpace datatype from the GMP protocol. GMP is a set of rules and datatypes for manipulating geometry for CFD applications. Only a brief overview is provided here, and more information can be found in [9]. Each ConfigSpace is defined by a set of Parameters (control surfaces), which are usually a single water-tight component (or group of components) in the surface triangulation of the geometry. These Parameters take either a finite number of discrete States, or a continuous range of values, which are specified as rigid-body motions relative to the complete Configuration. The allowable rigid-body motions are extremely general, so that no new application code is required to perform the geometry manipulation. The GMP specification is stored in an XML file, which can be parsed by the application control script to determine the number of Parameters, how many States are specified for each Parameter, etc. The application control script for the configuration space thus looks very similar to the velocity space script, except that the matrix of parameters are control surfaces and deflections, rather than changes to the velocity vector. The ConfigSpace datatype also allows the specification of groups of Parameters and States, for instance a coarse-, medium-, and fine-grained set of control surface deflections. A middleware layer between the GMP specification and the Cartesian applications actually does the work of translating the GMP Parameters and States into a unique geometric configuration.

2.3 Post-processing

One of the strengths of a modular process design is the ability to use 3rd-party tools when practical. These tools are used in many parts of the process in Fig. 1, e.g. PBS, GMP, and database software. The storage and search requirements for a CFD aerodynamic database are relatively modest, so a tree-based database storage solution is not a necessity, however the use of a standardized database does have benefits. Since database APIs are static and supported by a large user community, a CFD database can be portable across many application domains rather than focused on a niche. For example, a database with an SQL or ODBC interface can easily be ported to web services. Similarly, a supported API allows different domains, such as G&C, structures, and aerodynamics to speak a common language, and build their individual tools on a known stable system. Further, the use of a database software package allows the results of a CFD aerodynamic database to evolve to include items such as flowfield images, documentation, or cutting planes, which would be inconvenient to implement into a simpler format. SQLite[15] provides a nice compromise between a full client-server database implementation, and a self-contained ad-hoc file format. Rather than directly interface the post-processing applications to the database format, the database results are customized to the needs of the post-processing applications through a middleware layer. This approach again provides modularity and efficiency.

There are many post-processing applications which can utilize a CFD aerodynamic database: interactive flow visualization[16, 17], multi-disciplinary optimization, performance analysis, control law synthesis, etc. This focus of the current work is on the use of 6-DOF trajectories to develop and analyze G&C systems. The next two sections provide an overview of the algorithms and data structures required to fly a vehicle through a CFD database.

2.3.1 Multilinear Interpolation in d-Dimensions

The configuration and velocity spaces each contribute dimensions to the parameter space P of the study of any given configuration. For example, a study examining Mach, α, and β, with elevator, aileron and rudder deflections implies a 6-dimensional parameter space $P = P(M_\infty, \alpha, \beta, \delta_e, \delta_a, \delta_r)$. After building the aerodynamic database the aerodynamic coefficients are known at discrete locations in this parameter space. In order to perform trajectory simulations it is necessary to integrate through the parameter space using a 6-DOF package and control system model. At each timestep, the 6-DOF equations provide new conditions in the velocity-space, while the flight control system provides a new set of configuration parameters. Therefore, evaluating the aerodynamic coefficients for the vehicle at a new time level becomes a problem in d-dimensional interpolation, where d is the number of dimensions of P.

While a variety of approaches can be used for higher-dimensional interpolation, multilinear interpolation[18, 19] is an attractive choice for a first attempt. Without adaptive steering, the parameter space constitutes a regular Cartesian lattice in d-dimensions (a d-dimensional hypercube). The aerodynamics coefficients are known at every node of this Cartesian grid.

The sketch in Fig. 2 shows a 2-D projection of the interpolation problem. For any target point x_j with $j \in \{0, 1, ..., d-1\}$ we must find the bounding hypercube in P, and then interpolate from the known data at the corners of this hypercube. In d dimensions, the bounding hypercube will have 2^d corners. The i^{th} corner of of this cube has coordinates $z_i = (z_{i_0}, z_{i_1}, ...z_{i_{d-1}})$. The set of all such corners is $\mathcal{Z}$. For convenience, we can denote the vector of aerodynamic coefficients at this location as $f(z_i)$. If we map this bounding hypercube to the interval $[0,1]^d$, then multilinear interpolation gives an interpolated value at the target x_j.

$$\begin{aligned}\hat{f}(x) &= \sum_{i\in\mathcal{Z}} f(z_i) \prod_{j=0}^{d-1} \left(1 - \left|x_j - z_{i_j}\right|\right) \\ &= \sum_{i\in\mathcal{Z}} f(z_i) w_i(x)\end{aligned} \tag{1}$$

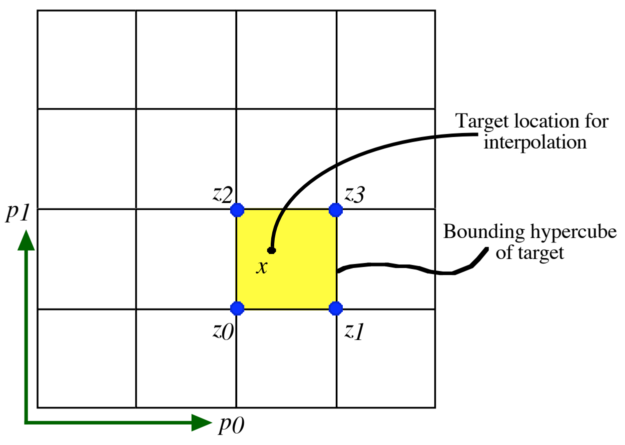

Target location for interpolation
Bounding hypercube of target
p1
p0
x
z0
z1
z2
z3

Figure 2: 2-D interpolation. In d dimensions this 2-D projection becomes a hypercube, with data known at the corners.

where w_i is the basis function associated with the i^{th} corner of the hypercube, and $w_i(x)$ is the weight of the i^{th} corner node in computing $\hat{f}(x)$. The basis function for multilinear interpolation is attractive since it has both d-fold symmetry, and a compact stencil. The summation in Eqn. 1 simply accumulates contributions from each corner of the hypercube, and hence the product is constant for each corner $z_i \in \mathcal{Z}$.

We find the bounding hypercube using successive binary searches in each of the d coordinate directions, then map the search result to a unit hypercube, and lastly evaluate the interpolation for all values of the aerodynamic coefficients (reusing the weights). Note that in very high dimensional spaces, evaluation of Eqn. 1 can potentially be expensive. The sum contains 2^d terms, each of which is itself a product over d terms. This gives an overall complexity of $\mathcal{O}(2^{d+1})$. Numerical tests suggest that modern floating-point chips give essentially instantaneous results even for parameter spaces with as many as 15 dimensions.

2.3.2 Controlled 6-DOF Integration

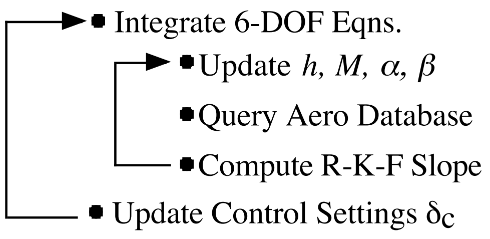
The general unconstrained motion of a rigid body is described by the Newton-Euler equations of motion, which break the motion into a translation of the center of mass, and a rotation about the body principal axis system. These 6-DOF equations are described in detail in the appendix to[13], along with verification examples, and only an overview of the features relevant to database traversal are discussed here. The GMP specification of 6-DOF motion[9], which has previously been used for coupled CFD/6-DOF simulations, is re-used here to specify the inputs for the database traversal. This allows automated tools to be developed for trajectory optimization or performance evaluation. A controller optimization example is included in Sec. 5.

Unlike solving the fully-coupled 6-DOF motion within a CFD simulation, where the evaluation of the forces and moments on the body requires an expensive CFD flow solution, a database traversal can evaluate the forces and moments efficiently at any time level using the interpolation procedure outlined in the previous section. This allows the use of high-order time-integration schemes for advancing the Newton-Euler equations. A 4th-order Runge-Kutta (R-K) scheme with adaptive timestep control provides a self-starting integrator along with an error estimator. With adaptive time resolution the integrated solution of each of the individual equations of motion satisfies an accuracy constraint, allowing automated methods to be developed. Here, Fehlberg's modified R-K scheme[20] (denoted by R-K-F) is used. Two integrated solutions are generated: one to 4th-order accuracy (y_4), and one to 5th-order (y_5). The difference between them provides an error estimate for the timestep ($e = |y_4 - y_5|$). If the error is below a predefined tolerance (e_{max}) the 4th-order solution is accepted, otherwise the procedure is repeated using a smaller timestep. The R-K-F scheme requires 6 evaluations of the forcing function for each timestep. This is in contrast to the common approach in adaptive timestep methods of recomputing the solution using half the original timestep and Richardson extrapolation to determine the error. For a 4th-order R-K scheme this timestep-halving method requires 11 evaluations of the forcing function at each step. The timestep is continuously and smoothly varied using the error estimate, even if the solution satisfies the accuracy constraint, according to

$$\Delta t^{n+1} = \Delta t^{n} \left(\frac{1}{2} \frac{e}{e_{max}} \right)^{1/p}$$

where $p = 4$ for a 4th-order scheme. This limits the number of times that the accuracy constraint is violated and also smoothly increases (decreases) the timestep in benign (stiff) regions, as the error estimate reacts to the slope of the solution. This approach is preferred over simpler halve/double/hold methods which halve or double the timestep when required/possible, and hold it fixed otherwise.

It is often desired to couple a controller to the 6-DOF integrator in order to direct the vehicle motion during the trajectory. In the current scheme the altitude and velocity vector are continuously updated within each R-K-F update step. After the solution is advanced to the next time level, the controller is called with the new state information, and the control settings are updated for the next timestep (Fig. 3). Future work will enhance this scheme by calling the controller at a fixed sampling interval to more closely model the actual system.



Integrate 6-DOF Eqns.
Update h, M, α, β
Query Aero Database
Compute R-K-F Slope
Update Control Settings δ_c

Figure 3: Coupled 6-DOF/controller update algorithm.

3 Validation of Aerodynamic Data - LGBB

Parametric studies on a 2nd-generation glide-back booster were conducted examining the effects of Mach, α and β variation. The geometry in this analysis is the Langley Glide-Back Booster (LGBB), a design for which there exists data from both experimental and Navier-Stokes analyses [3, 4]. The upper left frame of Fig. 4 shows a view of the geometry and Cartesian mesh. The vehicle has a cranked-delta planform, single tail, and canards. Since the model did not have movable control surfaces, this database only varies the velocity vector. Nevertheless, the presence of both independent experimental and simulation data over a wide range of parameters makes it useful for establishing the credibility of the Cartesian method for predicting aerodynamic performance. In total the aerodynamic database covered nearly 2900 flow conditions. The Mach number varied over $[0.2, 6.0]$, α over $[-5°, 30°]$, and $\beta = [0°, 4°]$.[*] The run matrix includes 38 separate Mach numbers, 25 angles of attack, and 5 side-slip angles, however not all Mach-α combinations were run for all values of side-slip angle.

Figure 4 gives an overview of the aerodynamic performance for this design. The Cartesian mesh shown in the upper left frame has approximately 1.4 million cells and was used for all subsonic and transonic cases. Supersonic cases used a smaller computational domain with approximately the same total cell count. The other frames in Fig. 4 contain carpet plots showing the variation of lift, drag, and pitching moment coefficients with variation of Mach number and angle-of-attack. The sideslip angle is fixed at $\beta = 0°$ in these plots. These figures all show mild behavior at supersonic Mach numbers with more non-linear behavior as the configuration passes through $M_\infty = 1$. The carpet plot of drag vs. Mach number and α shows the characteristic transonic drag rise through $M_\infty = 1$, a feature that is dramatically exacerbated with increasing angle-of-attack. Beyond Mach 1, the drag coefficient drops off as the shocks lay back in the accelerating flow in accordance with gas dynamic theory. Also as expected, pitching moment shows the most variation in the transonic flight regime as small changes lead to large movements of the shock system on the vehicle. Obviously, these carpet plots are only one slice through the full aerodynamic database (at $\beta = 0$), and

[*]The LGBB geometry possesses bi-lateral symmetry, thus it is unnecessary to compute negative sideslip conditions.

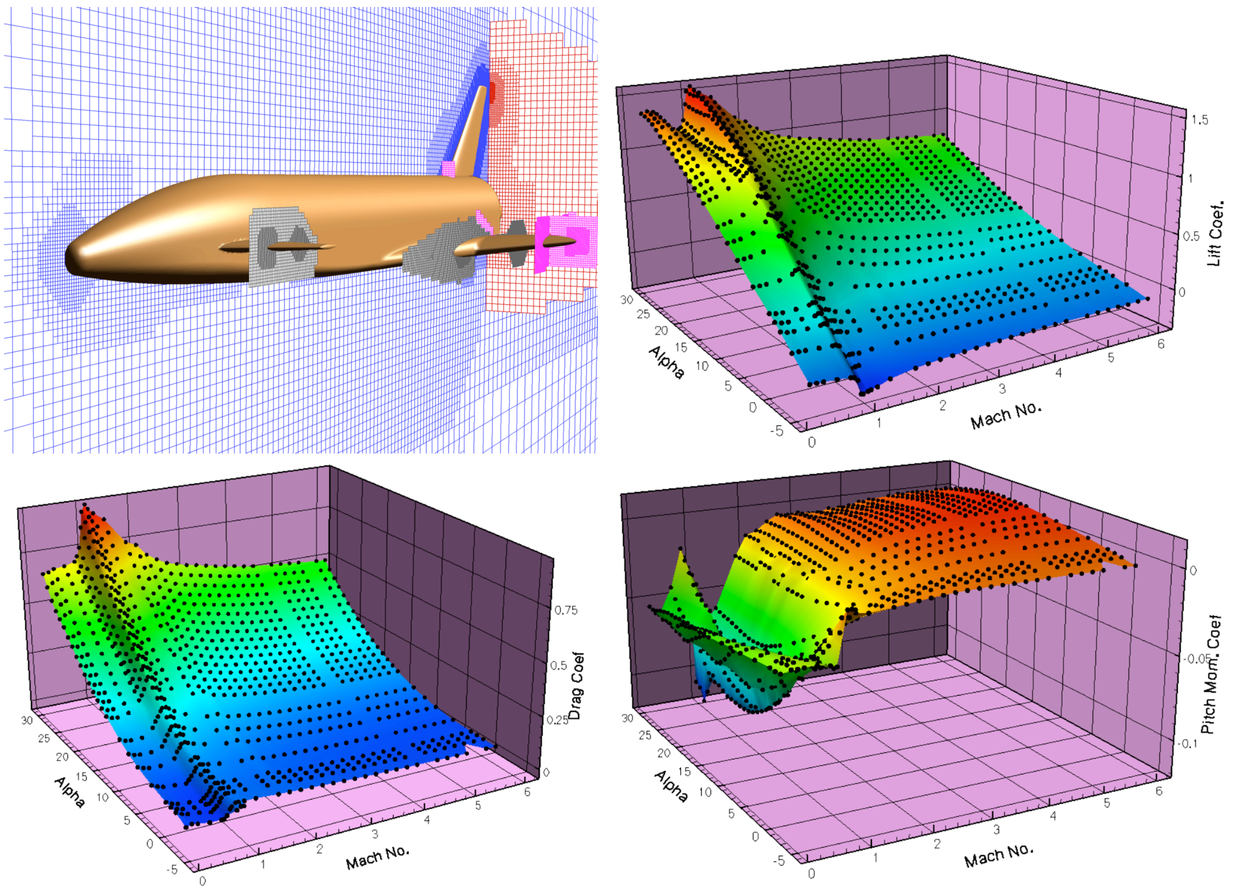


Figure 4: LGBB design with Cartesian mesh, and carpet plots showing the variation of lift, drag, and pitching moment with Mach number and angle of attack at $\beta = 0°$.

the data shown account for only about 20% of the cases run. Once the multi-dimensional aerodynamic database is computed, it is a straightforward task to examine the data using such lower-dimensional projections or slices. Nevertheless, visualization and feature detection in the full multi-dimensional dataset remains a research topic.

Figure 5 establishes confidence in the database through direct comparison with experimental data and Navier-Stokes simulations performed using NASA's Overflow solver[21]. This figure examines the variation of lift, drag, and pitching moment with angle-of-attack at Mach 1.6 and 2.0 while holding β fixed at 0 deg. Both simulation codes agree extremely well with the experimental data in lift and drag through to $\alpha = 30°$. Pitching moment for both simulation codes follows the data well to $\alpha = 15°$, and at higher angles of attack both codes over-predict the restoring moment. While Overflow does predict slightly less error in pitching moment at these high-angle-of-attack conditions, Cart3D produces an entire alpha sweep for the cost of a single Navier-Stokes solution. This accuracy and efficiency, combined with the automation of the Cartesian method for handling configuration changes, provides an excellent tool for building aerodynamic performance databases.

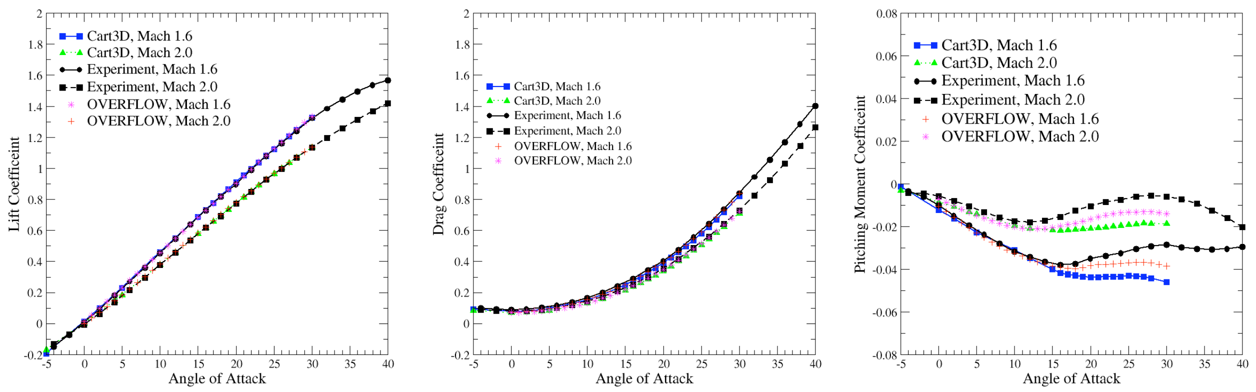


Figure 5: Comparison of lift, drag, and pitching moment for the LGBB with experimental data and Navier-Stokes results from [3] ($\beta = 0°$).

4 BEES Flyer Example

In this example, we demonstrate the use of both the velocity and configuration space components presented in Sec. 2 with a parametric study of an autonomous flyer. The bio-inspired engineering of explorations systems (BEES) flyer is envisioned as a small platform with sensing and control systems mimicking those of biological systems, for scientific exploration on the surface of Mars[10]. The current BEES flyer is a delta-wing with twin vertical tails and two elevons which provide pitch and roll control of the aircraft (cf. Fig. 6). Note that the outer-mold line of the flyer has not been optimized for a specific science mission on Mars, rather it is used as a generic test-bed for the development of adaptive flight control systems and sensors[10]. In order to evaluate the aerodynamic characteristics of the flyer, and to provide stability and control (S&C) information for the development of the flight control system, a database is computed over the expected flight domain. The flyer uses a prototype neural-network-based adaptive flight control system being developed at NASA Ames[22]. The CFD-generated aerodynamic data is used to validate

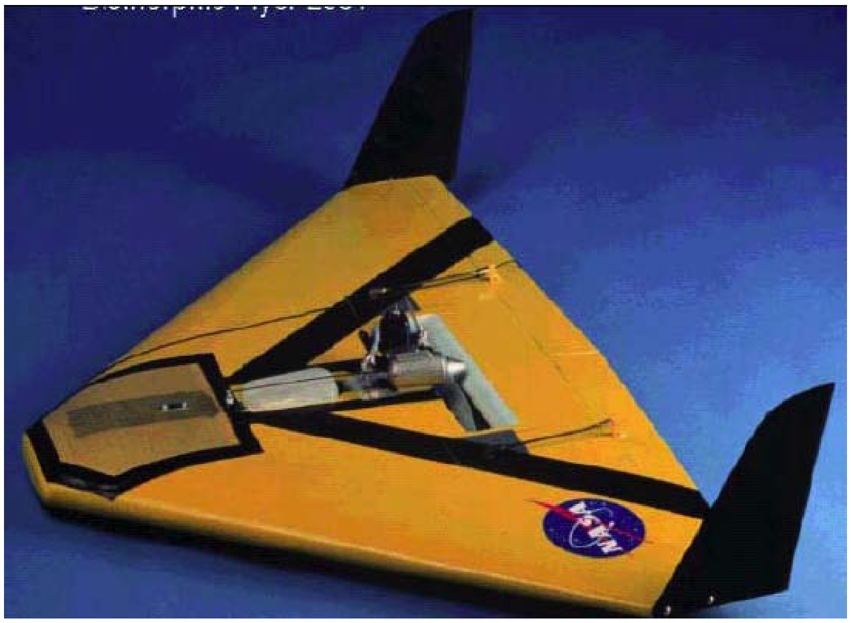


Figure 6: Prototype of the BEES flyer geometry (cf. Ref. [10] for more details).

and extend the controller developed using wind tunnel data.

The BEES flyer geometry was obtained from a CAD solid model[*], and is shown with a cutting plane through the computational mesh in Fig. 7. Each mesh contains approximately 1.5 million cells, with a finest refinement of 1.5 mm. As the elevons change position, the Cartesian meshing scheme responds to the change in surface definition and refines and coarsens the mesh appropriately. This causes the total number of cells in the mesh to vary depending upon the elevon settings.

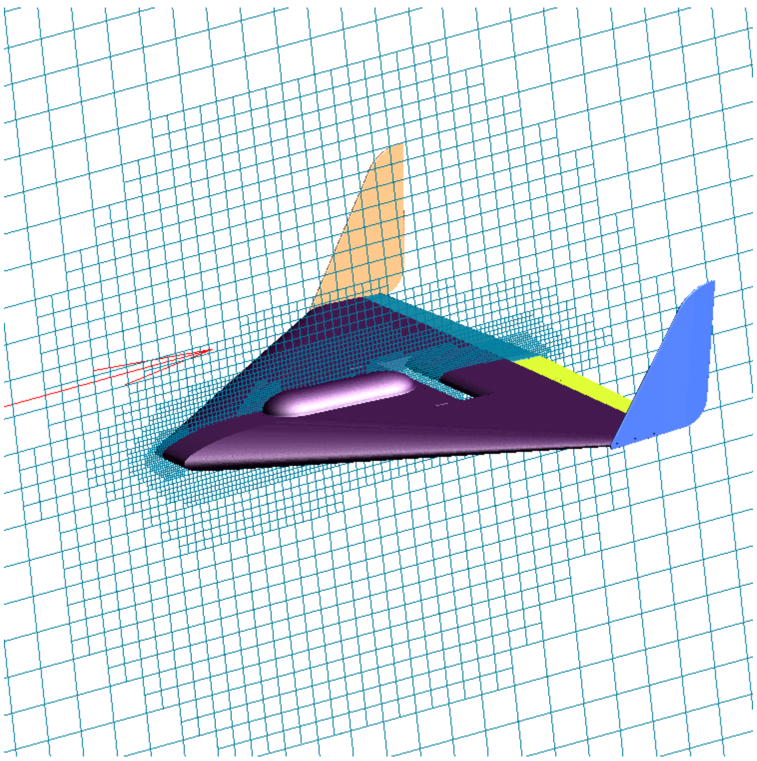


Figure 7: BEES flyer geometry with a cutting plane through the Cartesian mesh. The triangulated surface geometry was generated directly from a CAD solid model. Colors denote separate water-tight components of the configuration.

The aerodynamic database uses the following five parameters:

Configuration space :
Independent deflections of the left and right elevons. The deflection range is from 20 deg. trailing-edge down to 10 deg. up in 5 deg. increments resulting in 7 discrete settings for each elevon.

Velocity Space :
Mach numbers: 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8
Angle of attack: -10 to 15 deg. in 5 deg. increments
Sideslip angle: 0 deg. and -10 deg.

This combination of parameters results in 4704 independent analysis cases that include 49 configurations.

The results for the velocity space at the neutral elevon settings indicate that the performance of the present design is sufficient for subsonic flight at low angles of attack. However, the database also reveals that the drag-divergence Mach number is relatively low, roughly 0.7, and the drag rise quite severe. For the configuration space, Fig. 8 shows example results of the changes in roll and pitch as a function of different elevon deflection settings at $M_\infty = 0.6, \alpha = 5.0°$. The spheres denote discrete sample points. As expected, the aerodymanic coefficients behave smoothly and are consistent with the symmetry line of elevon deflection.

[*]The CAD model was provided by the Australian National University.

Further insight is provided by flow visualization, which is shown in Fig. 9. Here, 49 surface pressure distributions corresponding to the changing elevon deflection angles are shown at constant $M_\infty = 0.6, \alpha = 10.0°$. White regions in the figures denote supersonic flow. The influence of the propeller slot, as well as changes in the shock positions and aft-loading due to asymmetric elevon deflections is clearly seen. Furthermore, this flowfield visualization has also been used for identifying scientific sensor locations on the flyer to minimize interference from the flow.

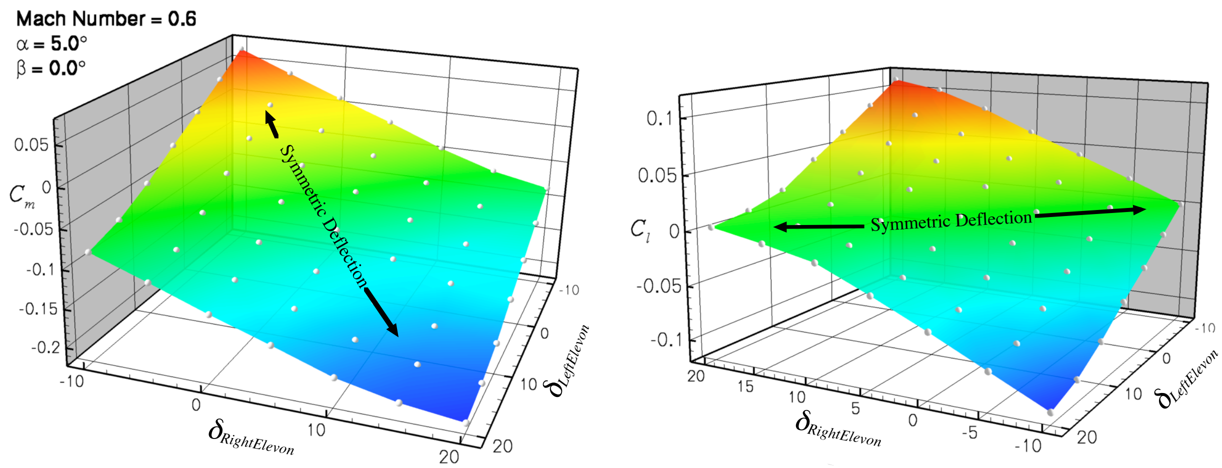


Figure 8: Variation of rolling and pitching moment with elevon deflection for the Mars flyer. The white spheres represent actual data-points. ($M_\infty = 0.6, \alpha = 5.0°, \beta = 0.0°$).

5 Space Shuttle Example

Even though NASA is actively engaged in the return-to-flight (RTF) initiative for the SSLV, both NASA and commercial groups are designing and analyzing the next generation of space-launch vehicles. A good example is the mission analysis for the LGBB concept of Pamadi et al. [23], which used engineering tools and wind tunnel data for the aerodynamic database. Before analyzing novel vehicles, the wealth of data associated with the SSLV can be used to develop and validate the existing methodology. As a first step in this effort, a simplified aerodynamic database for the SSLV in the ascent configuration is created with the automated Cartesian mesh toolset and used to develop a pitch controller.

5.1 Aerodynamic Database

The SSLV in the ascent configuration contains a number of active control surfaces: the left and right solid-rocket booster (SRB) engines, the 3 Space Shuttle main engines (SSMEs), and inner and outer elevons on each wing. These 9 control surfaces, along with the 3 components of the velocity vector lead to 12 independent parameters. If each of these parameters were given only 5 states it would lead to almost 250,000 datapoints, which is far more than

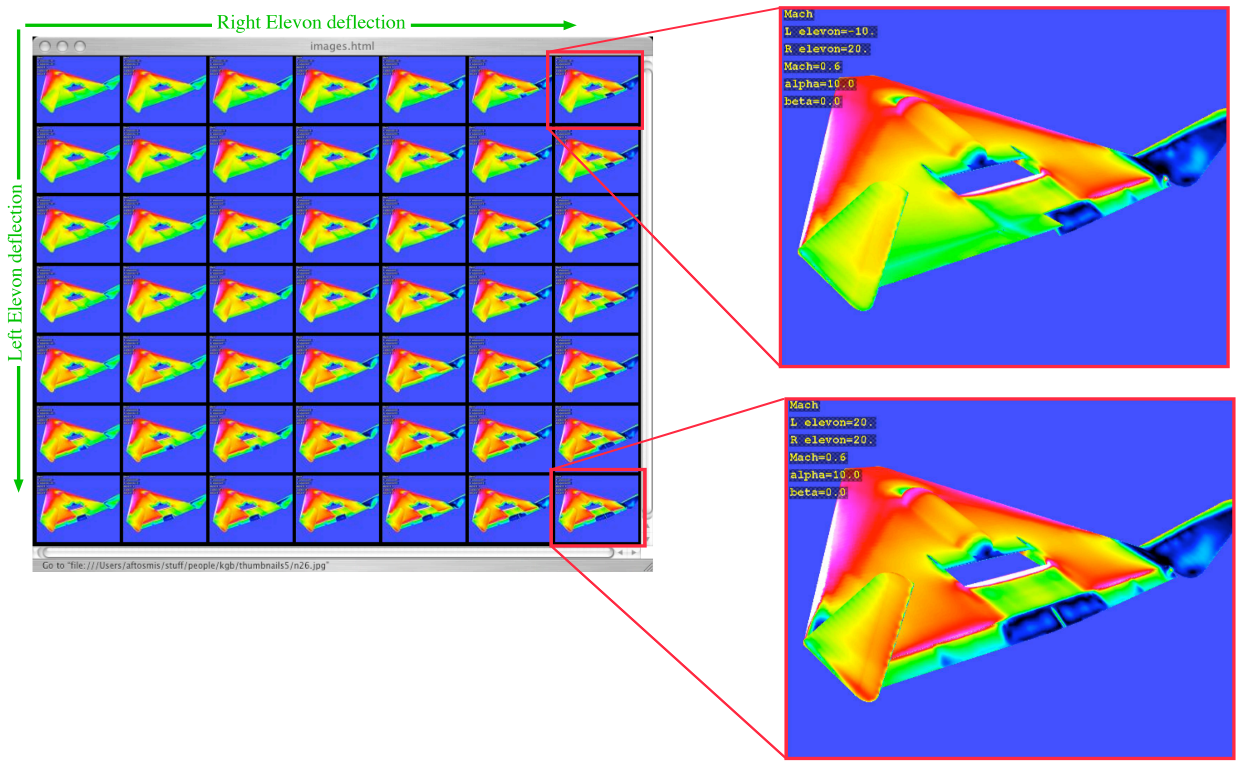


Figure 9: Surface pressure variation with elevon deflection for the Mars flyer. ($M_\infty = 0.6, \alpha = 10.0°, \beta = 0.0°$).

desired for an initial test. Since the elevons are used for wing load-relief during ascent, and not control, they are held fixed. Also, initially we are only interested in pitch control, so the left and right SRB engines and the left and right SSMEs can be treated as coupled. Further, the SSLV stack can be assumed to possess lateral symmetry. While this is not strictly true, the deviations are small and consistent with the desired level of modeling. This reduces the parameter space down to 3 control surfaces - the coupled SRB engines (δ_{SRB}), the center SSME (δ_c), and the coupled left and right SSMEs (δ_{lr}) - and 2 components of the velocity vector - Mach number and angle of attack. Since the current effort involves modeling the engine thrust, and engine performance varies with altitude, altitude is another independent parameter, albeit one that is loosely coupled to Mach number as the vehicle does not leave the launch pad at high Mach number. As a simplification, the altitude is chosen to be the median altitude over the Mach number range of interest. This Mach number/altitude correlation is determined from SSLV pre-flight tables. The simulated 6-DOF trajectories occur after the maximum q-loading condition the shuttle experiences, so all engines are at full power. The Mach number range is chosen as $M_\infty = [1.5, 3.5]$, which roughly corresponds to the period of maximum acceleration for the complete shuttle stack. The chosen angle-of-attack range is $[-4°, 4°]$.

An overview of the SSLV ascent configuration and a cutting plane through the Cartesian mesh is shown in Fig. 10. Prespecified adapation regions were supplied to resolve the engine plumes. Colors denote separate water-tight components of the configuration. Each mesh contains roughly 1.8 million cells for a half-body configuration. The major details of the geometry are modeled, including the fore and aft attach hardware, the liquid oxygen feedline, the stringers aft of the ogive section of the external tank (ET), separated wing elevons and tail rudders, the orbital maneuvering system pods, the body flap, and the details of the SRB and SSME nozzles.

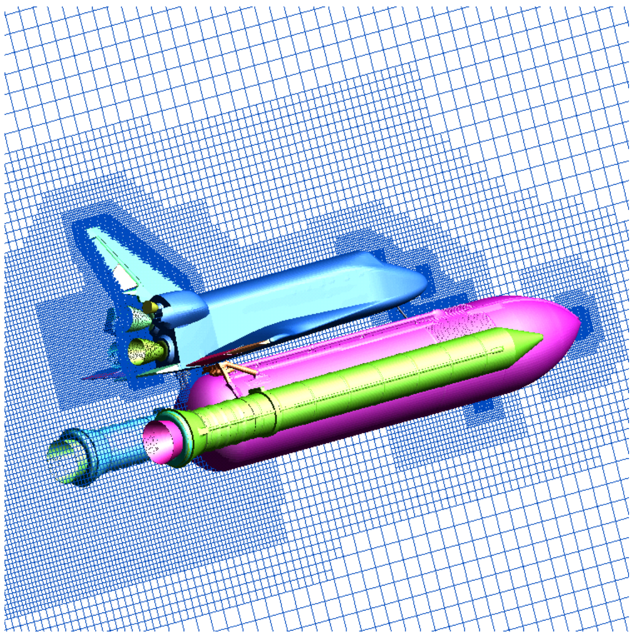


Figure 10: Cutting plane through the lateral symmetry plane of the Cartesian mesh for the SSLV. Prespecified adapation regions were supplied to resolve the engine plumes. Colors denote separate water-tight components of the configuration.

A full description and validation of the approach for simulating engine power settings within the Cartesian method is provided in a companion paper[24]. The details of the geometry for each engine nozzle is modeled from the combustion chamber to the exit nozzle. Figure 11 shows an SSME nozzle with partial cutaway to highlight the details of the internal nozzle geometry. Each engine nozzle uses a torus-shaped shroud geometry which connects the exterior surface of the nozzle to the SRB or orbiter. This shroud moves with the engine nozzle as it pitches and yaws. The SRB nozzles deflect $\pm 5^\circ$ in both pitch and yaw, and each SSME nozzle pitches $\pm 10.5^\circ$ and yaws $\pm 8.5^\circ$. The SRB engines are aligned with the longitudinal axis of the SRB in the zero-deflection orientation, while the center SSME nozzle is canted upwards 16° in the neutral orientation (cf. Fig. 11). The neutral position for the left and right SSMEs is rotated 10° upwards, but in the current test they are not canted outwards, even though 3.5° of yaw corresponds to the neutral position. The current unyawed orientation is sometimes referred to as a "pitch-lock" orientation for the SSMEs, and can be used for pitching maneuvers. The current work computes aerodynamic data with the SRB and SSME nozzles pitching over their full range of motion, however since we are only interested in pitch control, none are allowed to yaw. Plenum conditions are specified at the combustion chamber boundary as flow solver inputs for each engine. These conditions were supplied from real-gas simulations at the desired operating conditions[25].

Each of the 5 independent parameters is allowed to vary over 3 states (min., median, and max.), leading to 243 datapoints in the aerodynamic database. While this initially seems coarse, in the parameter range being examined most of the effects are expected to behave linearly, so that 3 states is sufficient, especially for a preliminary test. This can be seen in Fig. 12 which shows carpet plots of the normal force and pitching moment variation with pitch of the SRB and center SSME nozzles at $M_\infty = 2.5$, $\alpha = 0.0^\circ$. The aerodynamic

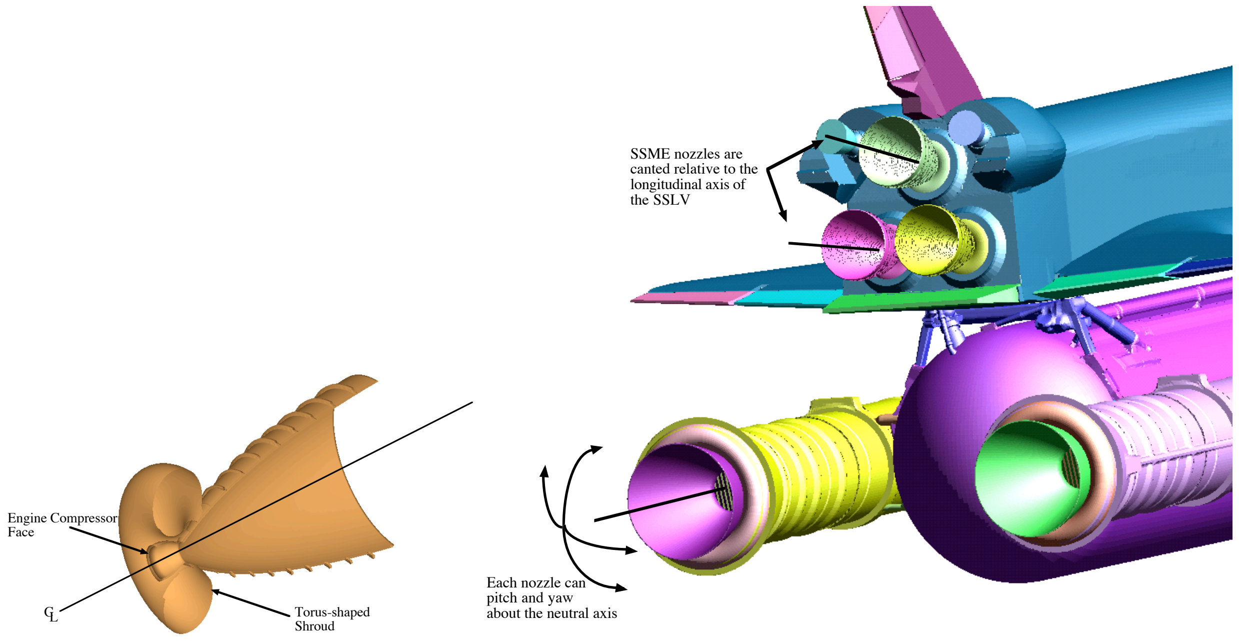
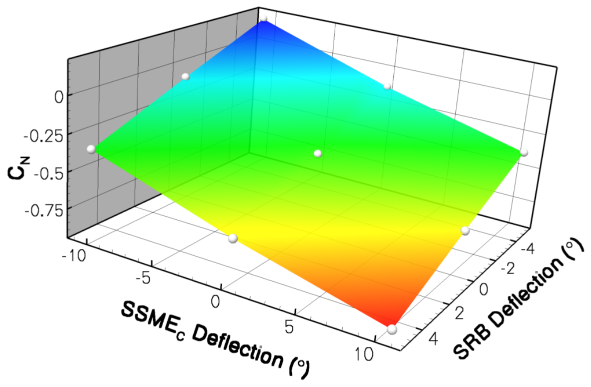

SSME nozzles are canted relative to the longitudinal axis of the SSLV
Engine Compressor Face
℄
Torus-shaped Shroud
Each nozzle can pitch and yaw about the neutral axis

Figure 11: Details of the SSME nozzle and orbiter aft end, showing the SRB and SSME gimbaling of the nozzles.

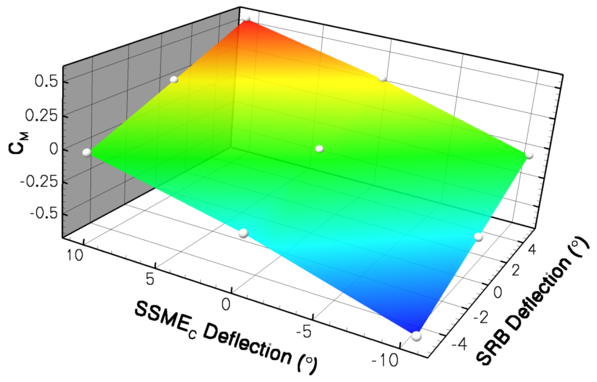
response at these conditions is relatively benign. The effect of SRB and center SSME pitch on the flowfield can be seen in Fig. 13, where a matrix of flow visualization pictures of Mach number contours on the lateral symmetry plane are presented at $M_\infty = 2.5$, $\alpha = 0.0°$. The plumes entrain most of the low-speed, separated flow behind the bluff afterbodies of the ET and orbiter. Interference between the center and left/right SSME plumes is visible. A normal shock forms below the mid-chord station of the orbiter wing at these flow conditions. Upstream of this location the contours are unchanged with nozzle deflection, consistent with a supersonic flowfield. After verifying and validating the current aerodynamic response and trajectory simulation tools, a refined database will be built using a general adaptive steering capability over a wider range of the parameter space.

5.2 Controller Optimization

The SSLV flies with an open-loop controller, using a table-lookup to determine the nozzle deflections, during the first stage of the ascent (while the SRBs are still attached). The current work attempts to develop a closed-loop feedback control system for the SRB and SSME pitch deflections using the computed aerodynamic database described in the previous section. The simulated trajectory is described in Fig. 14. The trajectory begins at a nominal mission elapsed time (MET) of 69 sec., at an altitude of 47,000 ft. and a Mach number of 1.75. The controller is commanded to maintain a constant angle of attack of 3° and flight path angle of 30°. The trajectory simulation is run for 27.5 sec., which



(a) Normal Force



(b) Pitching Moment

Figure 12: Aerodynamic response to SRB and center SSME nozzle deflections. White spheres represent actual datapoints, color corresponds to magnitude of dependent variable ($M_\infty = 2.5$, $\alpha = 0.0°$).

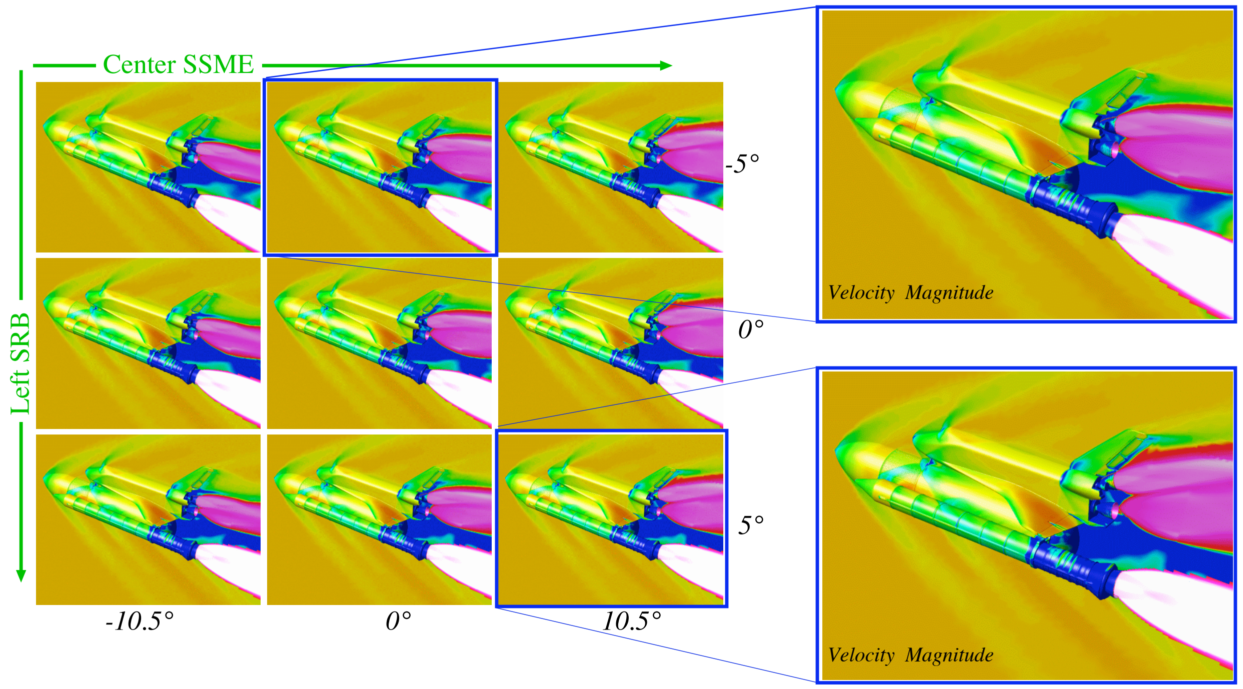


Figure 13: Changes in Mach number contours (red is high, blue is low) on the lateral symmetry plane in response to SRB and center SSME nozzle deflections ($M_\infty = 2.5$, $\alpha = 0.0°$).

roughly corresponds to the period of maximum acceleration for the complete shuttle stack.

The inertial properties of the SSLV were held fixed for the duration of the trajectory. This simplification ignores changes in the mass and moments of inertia due to fuel burn, and also the movement of the center of mass location due to fuel burn and “sloshing” of the liquid fuel in the ET. Models for these effects can be incorporated in a straightforward manner with the existing 6-DOF process if higher-fidelity trajectory results are required.

A proportional (P), integral (I), differential (D) controller (cf. Fig. 15) is used for each engine nozzle, according to

$$\delta = K_P e + K_I \int e dt - K_D \frac{de}{dt} \tag{2}$$

where e is the error between the command and measured angle of attack. Since the PID controllers for each nozzle are decoupled, the amplifier gains (K_P, K_I, K_D) form 9 independent variables (3 for each control surface nozzle) which define the control system. Manually determining the gains for a single-variable controller by trial and error, or using a model problem, is relatively straightforward, however for a complex multi-variable system such as the current one, manual tuning is prohibitive. The current example demonstrates the utility of trajectory simulations through a CFD aerodynamic database by coupling the 6-DOF database trajectory module with an optimization package to determine optimal values for the 9 controller gains for the specified mission.

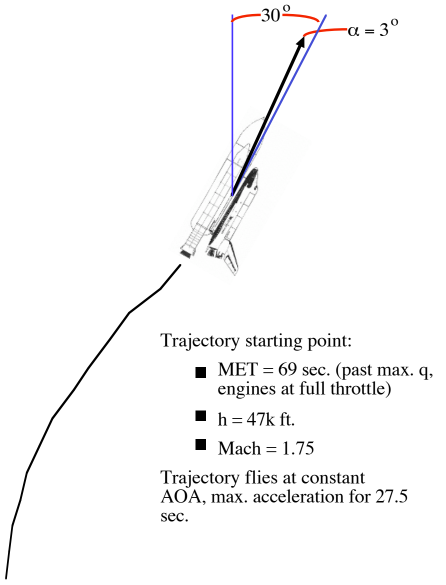

30°
α = 3°
Trajectory starting point:
MET = 69 sec. (past max. q, engines at full throttle)
h = 47k ft.
Mach = 1.75
Trajectory flies at constant AOA, max. acceleration for 27.5 sec.

Figure 14: Simulated trajectory for controller optimization.

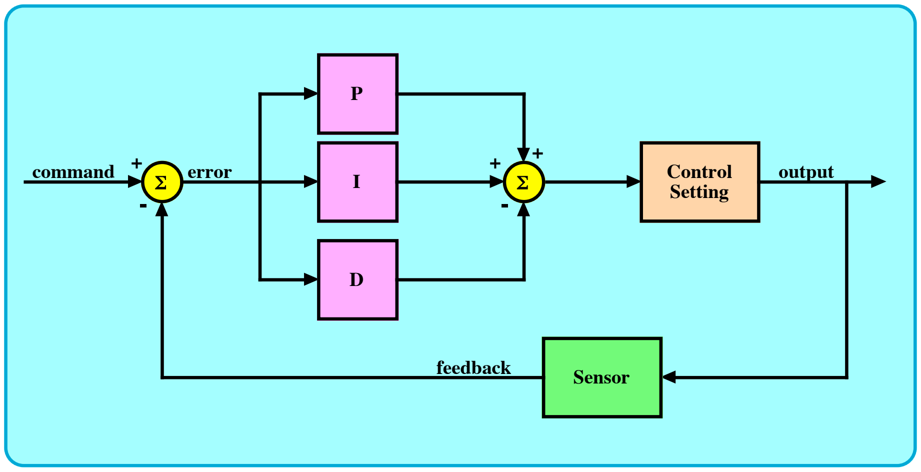


Figure 15: Schematic of a PID controller.

The optimizer uses a genetic algorithm (GA) developed by Holst and Pulliam[26], as the design space is likely multi-modal. Once the aerodynamic database is constructed, the cost of each trajectory simulation is essentially zero, so that the strength of a GA optimizer can be utilized. The selection operators of the GA (passthrough, average crossover, and mutation) were set to maintain a diverse population of candidate solutions and therefore avoid local optima. The objective function was chosen as the RMS-error of the angle of attack over the complete trajectory,

$$\mathcal{J} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\alpha(t_n) - \alpha_c\right)^2}$$

The convergence of the optimizer with GA evolution cycle is shown in Fig. 16, and the optimal controller gains are in Table 1. The K_I gain for the left/right SSME reached the minimum allowed value, which likely inhibits further convergence, though the resulting error is still very low.

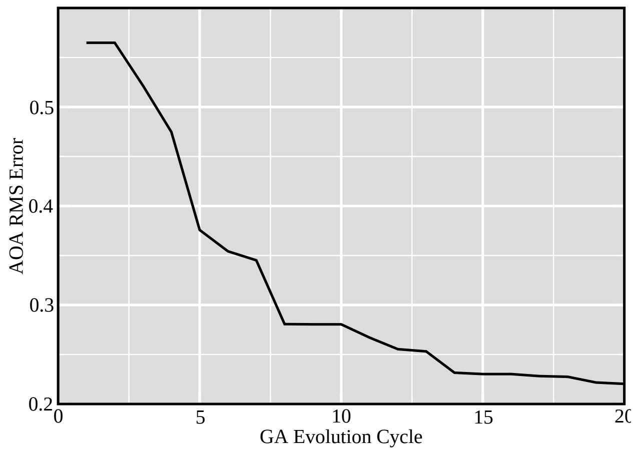


Figure 16: Convergence of the GA objective function.

Controller	K_P	K_I	K_D
δ_{SRB}	-48.51	128.3	13.42
δ_c	-94.78	16.80	26.07
δ_{lr}	-83.12	-100.0	-50.89

Table 1: Optimal controller gains computed for the trajectory described in Fig. 14.

The maximum error in angle of attack at any instance in the trajectory computed with the optimal controller gains is 0.015° (cf. Fig. 17). The initial trim position of the trajectory simulation appears to be inconsistent with the controller, hence initially the controller must correct this error. This also likely limits the convergence of the optimizer. The SRBs and center SSME operate in-phase, while the left/right SSME provide an out-of-phase counter-balance. This requires the left/right SSMEs to provide more power (greater pitch amplitude). The Mach number and altitude of the optimal simulated controller are compared to the STS-107 trajectory in Fig. 18. The STS-107 flight is close to a nominal SSLV ascent trajectory.

The simulated trajectory climbs faster than the STS-107 flight, though overall the comparison is favorable, especially for an initial qualified test.

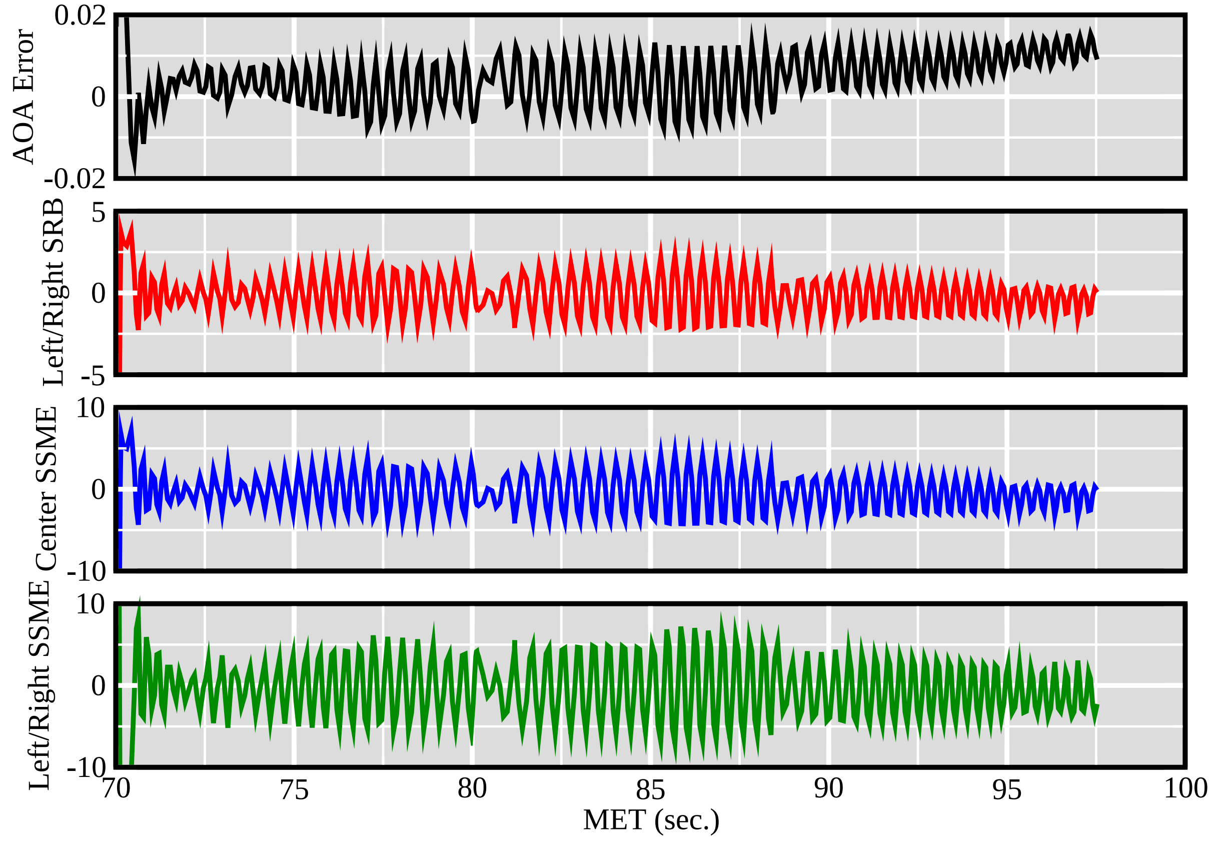
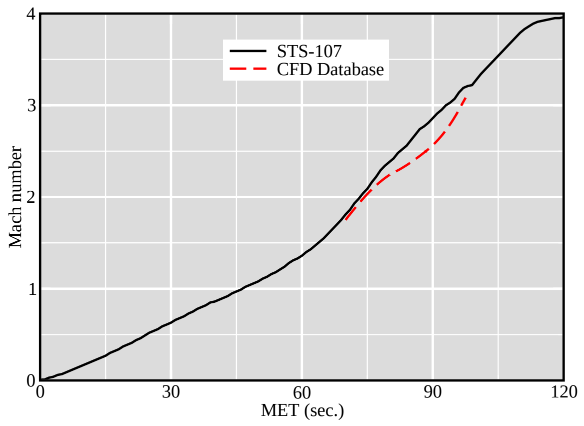


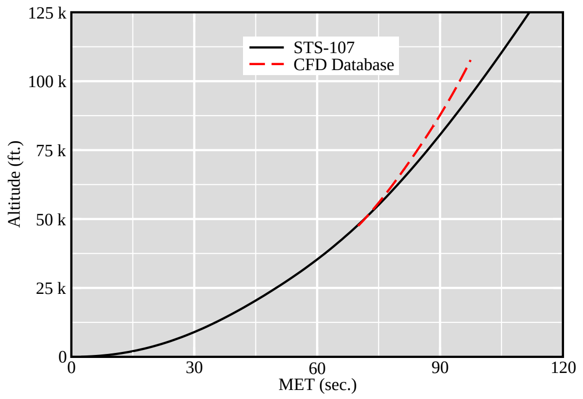
Figure 17: Error in angle of attack, and engine controller positions for the trajectory computed with the optimal controller gains.

6 Future Work

A modular process for performing parametric studies about a configuration using a Cartesian method has been described. This process leverages existing stand-alone applications for performing isolated steady-state simulations, and glues them together with control scripts to provide functionality for building aerodynamic databases. A novel part of this process is the automatic handling of general control surface deflections based upon simple, user-specified inputs. Several examples of building an aerodynamic database using the automated Cartesian CFD method were presented, both with and without moving control surfaces. These examples highlight the use of an aerodynamic database to develop and analyze G&C systems and performance requirements. With the basic infrastructure in place and tested, improvements can be initiated. These will initially involve two main areas: adaptive steering, and



(a) Mach number



(b) Altitude

Figure 18: Comparison of the trajectory computed with the optimal controller gains and the STS-107 trajectory.

validating and augmenting the SSLV aerodynamic database.

Steering is a critical component of developing an aerodynamic database as it directly impacts both efficiency, by optimizing the benefit for a fixed amount of work, and the accuracy by placing datapoints where there are abrupt changes in the S&C derivatives. This choice of a steering strategy is loosely coupled to the use of the database in analysis programs. A general adaptive steering capability requires an unstructured data format and complex interpolation algorithms. This data format capability will be developed and tested with various steering strategies using the example configurations examined in this paper.

The aerodynamic database built for the SSLV is an initial effort. The aerodynamic response will be validated against available data and comparison with Navier-Stokes simulations from the Overflow solver. With a validated process, refinements to the database and trajectory modeling can be pursued if necessary. The capability to automatically and efficiently generate any ascent configuration for the SSLV, including operating engine nozzles is an important part of ongoing RTF efforts. The current automated Cartesian method can be utilized in rapid-response, launch-day simulations, or other time-critical high-fidelity analyses.

Acknowledgments

The authors would like to thank Ray Gomez and Darby Vicker of NASA Johnson Space Center for their continued support and assistance. Scott Murman was supported by NASA Ames Research Center (contract NAS2-00062) during this work. This work was performed while Marian Nemec held a National Research Council Research Associateship Award at the NASA Ames Research Center.

References

[1] Yarrow, M., McCann, K.M., DeVivo, A. and Tejnil, E., "Production-Level Distributed Parametric Study Capabilities for the Grid," in *Proceedings of Grid 2001 2nd International Conference on Grid Computing*, 2001.

[2] Murman, S.M., Chaderjian, N.M., and Pandya, S. A., "Automation of a Navier-Stokes S&C Database Generation for the Harrier in Ground Effect," AIAA Paper 2002-0259, Jan. 2002.

[3] Rogers, S. E., Aftosmis, M. J., Pandya, S. A., Chaderjian, N. M., Tejnil, E. T., and Ahmad, J. U., "Automated CFD Parameter Studies on Distributed Parallel Computers," AIAA Paper 2003-4229, June 2003.

[4] Chaderjian, N.M., Rogers, S.E., Aftosmis, M.J., Pandya, S.A., Ahmad, J.U., and Tejnil, E., "Automated CFD Database Generation for a 2nd Generation Glide-Back-Booster," AIAA Paper 2003-3788, June 2003.

[5] Aftosmis, M.J., Berger, M.J., and Melton, J.E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, 36(6):952–960, June 1998.

[6] Haimes, R. and Aftosmis, M.J., "On Generating High Quality "Water-tight" Triangulations Directly from CAD," in *Meeting of the International Society for Grid Generation (ISGG) 2002*, June 2002.

[7] Aftosmis, M.J., Berger, M.J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA Paper 2000-0808, Jan. 2000.

[8] Aftosmis, M.J., Berger, M.J., and Murman, S.M., "Applications of Space-Filling-Curves to Cartesian Methods for CFD," AIAA Paper 2004-1232, Jan. 2004.

[9] Murman, S.M., Chan, W.M., Aftosmis, M.J., and Meakin, R.L., "An Interface for Specifying Rigid-Body Motion for CFD Applications," AIAA Paper 2003-1237, Jan. 2003.

[10] Soccol, D.D., Chahl, J.S., Le Bouffant, J., Garratt, M.A., Mizutami, A., Thurrowgood, S.A., Ewyk, G., and Thakoor, S., "A Utilitarian UAV Design for NASA Bioinspired Flight Control Research," AIAA Paper 2003-461, Jan. 2003.

[11] "Portable Batch System." http://pbs.mrj.com.

[12] Murman, S.M., Aftosmis, M.J., and Berger, M.J., "Implicit Approaches for Moving Boundaries in a 3-D Cartesian Method," AIAA Paper 2003-1119, Jan. 2003.

[13] Murman, S.M., Aftosmis, M.J., and Berger, M.J., "Simulations of 6-DOF Motion with a Cartesian Method," AIAA Paper 2003-1246, Jan. 2003.

[14] Nemec, M., Aftosmis, M.J., and Pulliam, T.H., "CAD-Based Aerodynamic Design of Complex Configurations Using Cartesian Methods," AIAA Paper 2004-0113, Jan. 2004.

[15] "SQLite." http://www.sqlite.org.

[16] "ParaVista." http://www.viz-solutions.com/ParaVista.htm.

[17] Sandstrom, T., Henze, C., and Levit, C., "The hyperwall," in *Proc. of IEEE International Conference on Coordinated & Multiple Views in Exploratory Visualization*, pp. 124–133, 2003.

[18] Burnett, David S., *Finite Element Analysis.* Addison-Wesley, Reading, MA, 1987.

[19] Hill, Steve, "Tri-linear interpolation," in *Graphics Gems IV* (Pual Heckbert, ed.), pp. 521–525, Academic Press, Boston, MA, 1994.

[20] Fehlberg, E., "Low-order Classical Runge-Kutta Formulas with Stepsize Control and Their Application to Some Heat Transfer Problems," NASA TR R-315, 1969.

[21] Jespersen, D.C., Pulliam, T.H., and Buning, P.G., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.

[22] Krishnakumar, K., Gundy-Burlet, K.G., Aftosmis, M.J., Nemec, M., Limes, G., Berry, M., and Logan, M., "Intelligent Control for the BEES Flyer," AIAA Paper 2004-6274, Sept. 2004.

[23] Pamadi, B.N., Tartabini, P.V., and Start, B.R, "Ascent, Stage Separation and Glideback Performance of a Partially Resusable Small Launch Vehicle," AIAA Paper 2004-0876, Jan. 2004.

[24] Pandya, S.A., Murman, S.M., and Aftosmis, M.J., "Validation of Inlet and Exhaust Boundary Conditions for a Cartesian Method," AIAA Paper 2004-4837, Aug. 2004.

[25] D'Agostino, M.G., Marshall Space Flight Center, Huntsville, AL, personal communication.

[26] Holst, T. L. and Pulliam, T. H., "Aerodynamic Shape Optimization Using a Real-Number-Encoded Genetic Algorithm," AIAA Paper 2001-2473, June 2001.